REMARKS

Claims 1-4, 6, 8-12, 14, 16-20, 22, and 24-32 are pending in the present application. Claims 1, 9, 17, and 29 are amended. Reconsideration of the claims is respectfully requested. Applicants thank the Examiner for the courtesy shown in making the current office action non-final.

I. 35 U.S.C. § 101

The office action rejects claims 1, 9, 17, 25, and 29 under 35 USC § 101 as being directed to non-statutory subject matter. The office action provides examples of how to overcome the rejection. The examples appear to indicate that the rejection is based on the assertion that these claims are directed to intangible computer program products. Applicants have amended the claims accordingly. Support for the amendments can be found on page 10, lines 23-27 and on page 13, line 28 through page 14, line 14 of the specification. Therefore, the rejection under 35 USC § 101 has been overcome.

II. 35 U.S.C. § 103, Obviousness

The office action rejects claims 1, 2, 4, 5, 7, 8, 10, 12, 13, 15, 16, 18, 20, 21, and 23-32 under 35 U.S.C. § 103 as being unpatentable over Borland Turbo Debugger User Manual (Borland) in view of Shagam, Source Code Debugging Tool, U.S. Patent 6,161,216 (Dec. 12, 2000). However, the office action appears to reject all pending claims, 1-4, 6, 8-12, 14, 16-20, 22, and 24-32. This rejection is respectfully traversed.

II.A The Proposed Combination Does Not Teach or Suggest the Invention of Claim 1

Regarding claim 1, the office action states that:

> Borland teaches the ability to build a Symbol Table externally is taught in the Borland Turbo Debugger. The file extension for the external symbol table is .TDS as on page 377 also see page 358 for the loading of the external table. The teaching of generating an external Symbol Table is on page 375. The Symbol Table in the debugger is loaded into memory Borland uses a Terminate and Stay Resident (TSR) routine (on first reading this might not be clear – be sure to read the memory mapping for symbols (pages 304 and 307) OR refer to prior art of record. It is the loading and

> comparison of the ST. The Borland reference provides evidence of this comparison when error messages are generated (page 378). In terms of options Borland teaches the ability to generate a Symbol Table (Option one – also includes external, import or no symbol table). In terms of the second option the option to run a debugger inherently uses a call graph which maps the possible execution paths. Shagam builds the "test script (Shagam, Abstract) from the call graph. The Applicant has used the terms "log" for these well known structures. Borland teaches a technique for building external Symbol Tables. External Symbol Tables are taught in the prior art and are deemed obvious regardless of the utility to build them because ST enable debuggers to identify the symbols in a program. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to use the admitted prior art C compiler option to generate an external ST.

Office Action of December 17, 2004, p. 5.

Regarding claim 1, the office action has failed to state a prima facie obviousness rejection because the proposed combination does not teach or suggest the claimed invention. Contrary to the office action statements, Borland does not teach or suggest any of the limitations of claim 1, as emphasized in the following reproduction of claim 1:

> 1. A method of debugging a software program whose execution flow is responsive at least in part to a first set of options, said method executed on a data processing system, said method comprising the steps of:
>
> *generating a first log file by executing said program with said first set of options, said first log file including an indication of all functions executed by said program during this first execution, said first log file stored in a computer readable medium;*
>
> *modifying at least one of the first set of options, and then generating a second log file by executing said program with said modified first set of options, said second log file including an indication of all functions executed by said program during this second execution, said second log file stored in a computer readable medium; and*
>
> *comparing said first log file with said second log file to debug the software program.*

The Examiner misapprehends Borland. Borland discusses a debugger program used with DOS-based machines. Borland does not generate a first log file by executing the program, said log file including an indication of all functions executed as claimed. Similarly, Borland does not generate a second log file as claimed. Hence, Borland also

does not teach or suggest comparing the first log file with the second log file as claimed. Shagam fails to cure the lack of disclosure in Borland. Shagam is directed to a method of automatically generating a script that automatically indicates where breakpoints (or trace points) will be located in a program to be debugged. Shagam does not generate a second log file and does not compare log files as claimed. Thus, the proposed combination does not teach or suggest the claimed invention. Accordingly, the office action has failed to state a prima facie obviousness rejection of claim 1.

The office action asserts otherwise, citing pages 358, 375, 377, and 378 of Borland and the abstract in Shagam. However, the reasoning supplied by the office action is logically insufficient to state a prima facie obviousness rejection. The office action appears to state that Borland teaches an external symbol table, that the Borland symbol table is loaded in resident memory and compared, ("It is the loading and comparison of the ST"), that Borland teaches that the error messages show evidence provides evidence of comparison of the symbol table, that Borland shows that a debugger inherently uses a call graph to map possible execution paths, that Shagam builds a test script from the call graph, that a call graph is a log as used by Applicants, that external symbol tables are deemed obvious, and that it would have been obvious to use the admitted prior art C compiler option to generate an external symbol table. However, the office action fails to state to what the external symbol table is compared and Borland provides no indication to what the external symbol table is compared. The office action fails to state how the symbol table is at all relevant to the claims and Borland provides no indication how the symbol table would be relevant to the claims. The office action fails to explain why a call graph is a log as used by Applicants. The office action fails to state how a compiler option is relevant to an option selected for an *executable* program or to the debugging of an executable program. The office action fails to state any reason why one of ordinary skill would be motivated to combine these unrelated matters. Thus, the statements made in the office action, when considered as a whole, are logically insufficient to state a prima facie obviousness rejection. Accordingly, the office action did not state a prima facie obviousness rejection.

Applicants address each of the references cited by the office action. Page 375 of Borland shows error codes for Borland's debugger. However, none of the error codes

discuss generating or comparing log files as claimed. Only two of the error codes are relevant, and they are as follows:

> **Program has invalid symbol table**
> The symbol table attached to the end of your program has become corrupted. Re-create an .EXE file and reload it.
>
> **Program has no symbol table**
> The program you want to debug has been successfully loaded, but it does not contain any debug symbol information. *You'll still be able to step through the program using a CPU window and examining raw data, but you will not be able to refer to any code or data by name.*
>
> To create a symbol table in Turbo Pascal (5.0 or later), turn on Debug|Standalone Debugging (or use the /v command-line option with TPC.EXE). If you're using one of Borland's C or C++ compilers, you must compile with /v and link your program with TLINK, using the /v option, in order to get debug symbol information. If you're using Turbo Assembler, assemble with /zi and link with /v.

Borland, pp. 375-6 (emphasis added).

A symbol table alone does not teach or suggest the claimed invention. Although a symbol table may be used by the claimed invention, the symbol table is irrelevant to the patentability of claim 1. Thus, Borland does not show or suggest the limitations of claim 1.

In addition, Borland clearly indicates that the symbol table is created during compiling of the program. Claim 1 is directed to a method of debugging an executable program. As claimed, a log file includes an indication of all functions executed by the program. The executable program is then modified with a second option, a second log file created, and then the log files are compared. Because the symbol table in Borland is created during compiling of the program and not as claimed, Borland does not teach or suggest the claimed methods.

Furthermore, the emphasized text in Borland shows that debugging still takes place by running through the program step-by-step. The advantage to the symbol table is that one will be able to refer to code or data by name; however, Borland indicates that one still steps through the program. Thus, Borland's debugger suffers from the very same problem solved by Applicant's claimed invention – the need to review code

manually. Although Borland had usefulness in 1991 in that it helped to identify bugs using a symbol table, Borland's teaching does not show or suggest creating a log of executed functions during execution of the program, modifying an option to the program, creating a second log of executed functions during a second execution, and then comparing the logs as claimed. Thus, Borland does not show or suggest anything similar to claim 1.

Continuing to analyze the citations made by the office action, Applicants turn to pages 377 and 378 of Borland. Again, Borland only describes error codes which do not show or suggest the claimed method. Only three error codes are relevant. Applicants analyze each in turn.

> **Symbol not found**
> You entered an expression that contains an invalid variable name. You may have mistyped the variable name, or it may be in some procedure or function other than the active one, or out of scope in a different module.

Borland, p. 377.

The fact that a symbol is not found does not show or suggest creating and comparing log files as claimed. Thus, Borland does not teach or suggest the limitations of claim 1.

The next error message is as follows:

> **Symbol table file not found**
> The symbol table file that you have specified does not exist. You can specify either a .TDS or .EXE file of the symbol file.

Borland, p. 377.

This section of text may indicate that a symbol table may be in the form of a file. However, that fact does not show or suggest creating and comparing log files as claimed. The fact that a symbol table may exist as a separate file does not provide any indication that a log file should contain an indication of all functions executed by a program to be debugged during execution. Likewise, Borland provides no indication that the program be modified with a second option. Likewise, Borland provides no indication that the log files should be compared. Thus, Borland does not show or suggest any of the limitations of claim 1.

The next error message is as follows:

> **Unknown symbol**

> You entered an expression that contained an invalid local variable name. Either the module name is invalid, or the local symbol name or line number is incorrect.

Borland, p. 378.

Again, the fact that a symbol is unknown does not provide any teaching or suggestion regarding the limitations of claim 1. Because Borland does not show or suggest the limitations of claim 1, and because Shagam fails to cure Borland's lack of disclosure in this regard, the combination of Borland and Shagam does not teach or suggest the claimed invention. Accordingly, the office action has failed to a state prima facie obviousness rejection of claim 1.

The office action also refers to page 358 of Borland for "the loading of the external table." Borland, which appears to describe a dialog box, is as follows:

> **Enter symbol table name**
> Enter the name of a symbol table to load from disk. Usually these files have an extension of .TDS. You must explicitly supply the file-name extension.

Borland, p.358.

This portion of Borland may show that a symbol table may be in the form of a file. As argued above, this fact does not teach or suggest the limitations of claim 1. Shagam fails to cure the lack of disclosure in Borland. Thus, the proposed combination does not teach or suggest the claimed invention and the office action has failed to state a prima facie obviousness rejection.

The office action also refers to the abstract in Shagam, which is as follows:

> A method and apparatus for debugging the source code using the source code debugger includes the following steps. A script generator is provided to receive source code instructions. Executing the script generator includes reading each source code instruction and generating, based on the type of instruction, a debugging script. The debugging script includes a specification of trace points. The debugging script is then provided to the source code debugger.

Shagam, Abstract.

Contrary to the assertions made in the office action, Shagam does not show a log as claimed. Shagam's script is a list of breakpoints or trace points automatically generated by Shagam's method, as shown by the following text in Shagam:

> The invention features a script generator **20** which automatically inserts the trace points within source code **10** and generates a debugging script **22** *that specifies where a trace point is to be inserted and the particular information to be collected.* Debugging script **22** is then provided to source level debugger **16** which, together with a symbol table **24**, translates symbolic information into absolute addresses. Symbol table **24** (generated by compiler/linker **12**) is used to make connections between source code **10** and the machine code that is generated by compiler/linker **12**.

Shagam, col. 3, ll. 51-60 (emphasis added).

The emphasized text in Shagam shows that his debugging script specifies where a trace point is to be inserted and also specifies the information to be collected. The point of Shagam's invention is that the script is automatically generated.

In contrast, claim 1 requires that the log files be generated by executing the program to be debugged and that the log files include an indication of all functions executed by the program during execution. Shagam's script is not generated in this manner. Claim 1 also requires that the log files be compared to debug the software program. Shagam provides no teaching or suggestion that separate log files be compared where one log file is generated after changing an option of the executable program to be debugged as claimed.

As shown above, Borland does not teach or suggest the limitations of claim 1. Shagam is also does not show or suggest the limitations of claim 1. Thus, the proposed combination does not teach or suggest the claimed invention. Accordingly, the office action has failed to a state prima facie obviousness rejection of claim 1.

II.B <u>The Office Action Failed to State a Motivation to Combine the References to Achieve the Invention of Claim 1</u>

The office action failed to state a prima facie obviousness rejection because the office action failed to state a motivation to combine the references. The office action does state that:

> External Symbol Tables are taught in the prior art and are deemed obvious regardless of the utility to build them because ST enable debuggers to identify the symbols in a program. Therefore, it would have been obvious to one of ordinary skill in the art at the

time of invention to use the admitted prior art C compiler option to generate an external ST.

Office Action of December 17, 2004, p. 5.

Under the standards of Graham v. John Deere, the Examiner must state a motivation to combine the references. In other words, the rejection must show reasons that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for combination in the manner claimed. *In re Rouffet*, 149 F.3d 1350, 47 USPQ 2d 1453 (Fed. Cir. 1998). "[w]hen determining the patentability of a claimed invention which combines two known elements, 'the question is whether there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination.'"

A statement that a purported fact is "deemed" obvious, without further support, is insufficient to provide a motivation to combine the references. Furthermore, the office action concludes without sufficient support that it would have been obvious to use the admitted prior art C compiler option to generate an external ST. Even if the first statement somehow inferred the second statement, the office action has only stated a fact unrelated to the claims. A compiler option is irrelevant to an option that affects execution flow of an executable program as claimed. Because neither office action statement provides a motivation to combine the references, the office action has failed to state a prima facie obviousness rejection.

In addition, an external symbol table is not claimed. What is claimed is generating a first log, a second log, and comparing the logs as claimed in claim 1. As argued above, generating an external symbol table in no way teaches or suggests the claimed limitations. Accordingly, the office action has again failed to state a prima facie obviousness rejection.

In addition, Borland shows a debugging program used in DOS machines in which symbol tables are used to identify and name data and code while debugging a program line-by-line. Shagam shows a method of automatically generating a script that includes information regarding where breakpoints should be inserted in an executable program and regarding what data to collect. No one would combine the references because the result would be an executable program that would have breakpoints automatically

inserted but that would have to be read line-by-line. In modern debugging methods, no one would revert to the old methods shown by Borland because doing so is slow, tedious, error-prone, and cost ineffective. Thus, neither Borland nor Shagam teach or suggest the limitations of claim 1 and the combination of Borland and Shagam do not teach or suggest the limitations of claim 1. Accordingly, the office action has failed to state a prima facie obviousness rejection of claim 1.

II.C The Examiner Used Impermissible Hindsight When Fashioning the Rejection

As shown above, the office action statements are logically insufficient to state a prima facie obviousness rejection. Furthermore, neither reference teaches or suggests the claimed inventions. Therefore, the Examiner must have used Applicants' own disclosure as a template to fashion the obviousness rejections, which is impermissible hindsight. Because the Examiner used impermissible hindsight, the office action has failed to state a prima facie obviousness rejection.

II.D Claim 1 is Non-Obvious Because No One Would Be Motivated to Combine the References

As argued above, the proposed combination would result in a backwards step in the art of debugging programs. Although Shagam would automatically create a script that shows where to insert breakpoints, Borland still suggests that the program be debugged by reading the program line-by-line. No one would do this today or at the time this application was filed. Therefore, no motivation exists to combine the references. For similar reasons, Borland teaches away from the claimed invention. Accordingly, claim 1 is non-obvious.

In addition, because Borland's symbol tables and Shagam's symbol tables are generated prior to program execution, and in particular are generated as part of a pre-execution program compilation operation, there would have been no ability to include function execution status as a part of Borland's symbol table. Thus, there would have been no motivation to modify Borland's symbol tables in accordance with the invention recited in claim 1.

II.E Claim 1 is Non-Obvious Because No One Would Be Motivated to Modify the Proposed Combination Further

As argued above, the proposed combination does not teach or suggest the invention of claim 1. In addition, neither reference provides any teaching or suggestion to modify the references further to achieve the claimed invention, especially in the light that the claimed invention greatly depart from the teachings of both references. No indication exists that separate log files be created based on execution of the program to be debugged and then comparing the log files, as claimed. Thus, when the references are considered as a whole, one of ordinary skill would not be motivated to modify the proposed combination further. Accordingly, claim 1 is non-obvious.

II.F Claims 2-4, 6, 8-12, 14, 16-20, 22, and 24

Independent claims 9 and 17 both contain limitations similar to those present in claim 1. Thus, the office action failed to state a prima facie obviousness rejections of these claims for the reasons given above. In addition, claims 9 and 17 are non-obvious for the reasons given above. Thus, independent claims 1, 9, and 17 are allowable over the cited references. Hence, claims 2, 3, 6, 8, 10-12, 14, 16, 18-20, 22, and 24, which depend on claims 1, 9, and 17 accordingly, are also allowable over the cited references.

In addition, these dependent claims contain other patentable features not taught or suggested by the cited references. For example, neither reference shows or suggests generating said first log file including a first set of return codes as claimed in claim 2. Thus, the dependent claims are non-obvious.

II.G Claims 25-32

The office action rejects claim 25 as obvious over Borland and Shagam under the same assertion with regards to claim 1. However, like claim 1, claim 25 requires generating a file including said listing of executable functions. As shown above, neither reference shows or suggests this step. Thus, the office action failed to state a prima facie obviousness rejection of claim 25 because the proposed combination does not teach or suggest the claimed inventions. Similarly, the office action failed to state a prima facie

obviousness rejection of claim 25 because the office action failed to provide a motivation to combine the references. Similarly, claim 25 is non-obvious because no one would be motivated to combine the references for the reasons given above. Thus, claim 25 should be allowable over the cited references. Claims 26-32 depend from claim 25 and therefore should be allowable by virtue of their dependence.

II.H Summary

The office action failed to state prima facie obviousness rejections because the proposed combination does not teach or suggest the claimed inventions, because the office action failed to state a motivation to combine the references, and because the reasoning stated in the office action is logically insufficient to state a prima facie obviousness rejection. In addition, the claims are non-obvious because no one would be motivated to combine the references. Therefore, the rejection of claims 1-4, 6, 8-12, 14, 16-20, 22, and 24-32. under 35 U.S.C. § 103 has been overcome.

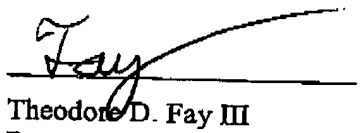
III. Conclusion

It is respectfully urged that the subject application is patentable over Borland and Shagam and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 17, 2005

Respectfully submitted,



Theodore D. Fay III
Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants